PROJECT REPORT

ON

COMPUTER GRAPHICS

“Feed the snake”

Carried By

**Mujammil Ansari**

**Varun Kalwar**

**Amaan Modak**

Project report submitted in fulfillment of the degree in

COMPUTER ENGINEERING

Under the supervision of

**Ms. Arti Ambokar**

Department of Computer Engineering

2012-2013

**ACKNOWLEDGMENT**

This report is on the mini project “Feed the Snake” of the subject Computer Graphics.

Our team would humbly appreciate and thank everybody those who have lend their handful experiences and aid to complete this virtuous project. We would like to thank our Head Of Department of Computer Engineering, Professor Dinesh Deore for his righteous and respectable support. We would also like to thank our subject teacher Ms. ArtiAmbokar for her guidance and relentless help.

**Mujammil Ansari**

**Varun Kalwar**

**Amaan Modak**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| Sr.No | Topic | Page No. |
| 1. | Objective | 4 |
| 2. | Introduction | 5 |
| 3. | Specifications | 7 |
| 4. | Algorithm | 8 |
| 5. | Flowchart | 13 |
| 6. | Program Code | 21 |
| 7. | Functions Used | 29 |
| 8. | Output | 33 |
| 9. | Bibliography | 36 |

**OBJECTIVE**

The goal is to implement a simple yet complete version of the game “FEED THE SNAKE”. In the game the player directs a constantly moving snake with the 'K' (left), 'H' (up), 'P' (down), 'M' (right) keys. The player must successfully navigate the snake so that it avoids obstacles such as walls and it's own tail while it the continues it's never ending pursuit of candy. When the snake eats a candy (the head of the snake touches a candy) a new candy is placed randomly on the game board and the snake gets longer. Once the snake eats the candies the length of the snake and the score increases.

The game is actually divided into three levels. The first level begins when the game starts. The second level starts when the player reaches the score 100. The third and the final level begins when the player scores 200. When the player correctly guides the snake through score of 300 the player wins. If at any time the head of the snake touches either wall or the snake’s tail the player loses.

**INTRODUCTION**

Feed The Snake is a classical arcade game. You must control the snake who should eat a little for the travel. The object of the game is eat, eat and eat. From the little worm you can grow to the mighty python. We can advise the game for everyone who loves to play classical snake game.

As in the classic snake game you shouldn't cut into the walls. All that you want from the classical snake game with wide game possibilities is done in the Feed the Snake. Begin this snake game one more time and Feed the Snake become the game you will always play.

This is a small turbo c++ program of snake game using some *graphics.h* functions. Anyone who is familiar with C++ graphics programming can do it very easily.

Code Explanation

In the beginning of code explanation the first important part is initializing the graphics mode in turbo C to do drawing a rectangle and that is must be done before using every graphics related functions. This graphics initialization is done by a predefined function called *initgraph()* with a tiny code part given below :

int gdriver = DETECT, gmode, errorcode;

initgraph(&gdriver, &gmode, "C:\\TC\\BGI");

If your TC folder is in D drive than just type *"D:\\TC\\BGI"* instead of

*"C:\\TC\\BGI"* inside the *initgraph( )* function. So if you want to use graphics functions in your turbo c program, than you have to always write these two statement in the *main()* function.

If you initialize the graphics mode than, all *graphics.h* function can be used in the program. But if you need to come back to the normal text mode than you have to shut down the graphics mode by using the *graphics.h* function *closegraph( )*.

In this program we have use user-defined functions given below :

int moveXRight();

int moveXLeft();

int moveYUp();

int moveYDown();

void store(void);

void erase(void);

void init(void);

void test(int x);

void putAnObject(void);

void getAkey(void);

int score(void);

void gameOver();

void restart();

**REQUIREMENT SPECIFICATIONS**

1. **User Characteristics**
   * User should be proficient in C/C++ and Windows environment.
   * User should know to compile and run files in C/C++.
2. **Hardware Requirements**
   * Intel Pentium Processor
   * Minimum 32 MB RAM
   * Maximum 100 MB of Hard Disk free space
   * VGA compatible Monitor (CRT or LCD-TFT)
   * A Keyboard for input
3. **Software Requirements**
   * MS-Dos or Microsoft Windows Operating System
   * C/C++ Compiler as Language tool
   * A compatible Graphic Driver

**ALGORITHM**

1. START
2. Initialize the following as :
   1. length = 50 (i.e. initial length of the snake)
   2. snakeColor = RED, borderColor = YELLOW
   3. Three yellow rectangular walls separated by 5 pixels each drawn using

rectangle(100,100,540,380);

rectangle(105,105,535,375);

rectangle(110,110,530,370);

* 1. speed = 12 (i.e. initial moving speed of snake)
  2. score = 0, print “LEVEL 1”
  3. N = 7999 (i.e size of the snake)
  4. stIndex=0, dlIndex=0
  5. xPos = 320, yPos=20

1. Call function putAnObject()
   1. create an object as

objX = random(41) + 12;

objY = random(41) +12;

temp = getpixel(objX\*10, objY\*10);

if(temp ! = snakeColor)

circle(objX\*10, objY\*10, 4);

1. Call function getAkey()
   1. Input a key from keyboard in an infinite ‘for loop’ as

char key = getch();

* 1. if (key == ‘M’)

xPos = moveXRight();

go to Step 5.1

* 1. if (key == ‘K’)

xPos = moveXLeft();

go to Step 5.2

* 1. if (key == ‘H’)

yPos = moveYUp();

go to Step 5.3

* 1. if (key == ‘P’)

yPos = moveYDown();

go to Step 5.4

* 1. if (key == ‘q’ || key == ‘Q’)

exit(1);

go to Step 9

1. Move the snake as input from keyboard (i.e RIGHT, LEFT, UP, DOWN)
   1. To move RIGHT, do the following 800 times
      1. Call the function store() which stores the position of the snake
      2. circle (xPos++, yPos, 4);
      3. delay(speed);
      4. call the function erase() which erases the end trails of the snake as it moves
      5. call test(1);

go to Step 6.1

* + 1. if (xPos % 10 == 0)

if(khbit())

break;

return xPos;

go to Step 4

* 1. To move LEFT, do the following 800 times
     1. Call the function store() which stores the position of the snake
     2. circle (xPos--, yPos, 4);
     3. delay(speed);
     4. call the function erase() which erases the end trails of the snake
     5. call test(2);

go to Step 6.2

* + 1. if (xPos % 10 == 0)

if(khbit())

break;

return xPos;

go to Step 4

* 1. To move UP, do the following 800 times
     1. Call the function store() which stores the position of the snake
     2. circle (xPos, yPos--, 4);
     3. delay(speed);
     4. call the function erase() which erases the end trails of the snake as it moves
     5. call test(3);

go to Step 6.3

* + 1. if (xPos % 10 == 0)

if(khbit())

break;

return yPos;

go to Step 4;

* 1. To move DOWN, do the following 800 times
     1. Call the function store() which stores the position of the snake
     2. circle (xPos, yPos++, 4);
     3. delay(speed);
     4. call the function erase() which erases the end trails of the snake as it moves
     5. call test(4);

go to Step 6.4

* + 1. if (xPos % 10 == 0)

if(khbit())

break;

return yPos;

go to Step 4

1. test (int x)
   1. When moving RIGHT

tst = getpixel (xPos+4, yPos);

if (xPos == objX\*10 && yPos == objY\*10)

a = score(); which keeps the current score of the game

call level(a);

go to Step 7

if(tst == borderColor || tst == snakeColor)

call gameOver();

go to Step 8

* 1. When moving LEFT

tst = getpixel (xPos-4, yPos);

if (xPos == objX\*10 && yPos == objY\*10)

a = score(); which keeps the current score of the game

call level(a);

go to Step 7

if(tst == borderColor || tst == snakeColor)

call gameOver();

go to Step 8

* 1. When moving UP

tst = getpixel (xPos, yPos-4);

if (xPos == objX\*10 && yPos == objY\*10)

a = score(); which keeps the current score of the game

call level(a);

go to Step 7

if(tst == borderColor || tst == snakeColor)

call gameOver();

go to Step 8

* 1. When moving DOWN

tst = getpixel (xPos, yPos+4);

if (xPos == objX\*10 && yPos == objY\*10)

a = score(); which keeps the current score of the game

call level(a);

go to Step 7

if(tst == borderColor || tst == snakeColor)

call gameOver();

go to Step 8

1. level( int x)
   1. if( x == 100)

print “LEVEL 2”

speed=speed-3 increases the speed by decreasing the delay time

* 1. if( x == 200)

print “LEVEL 3”

speed=speed-3 further increasing the speed of the snake

* 1. if( x == 300)

print “YOU WIN”

exit(1)

go to Step 9

1. gameOver()
   1. print “GAME OVER”
   2. call restart() to start the game again

go to Step 2

1. STOP

**FLOWCHART**

1. General Flow Chart

2) moveXRight() 3) moveXLeft()



4) moveYUp() 5) moveYdown



6) getAKey() 7) store()



8) gameOver()



9) test(int x)

10) erase() 11) score()



12) putAnObject() 13) level(int x)



14) init(void) 15) restart()



**PROGRAM CODE**

#include<graphics.h>

#include<stdlib.h>

#include<math.h>

#include<stdio.h>

#include<conio.h>

#include<dos.h>

#include<iostream.h>

class Snake

{

public :

Snake(int x, int y){xPos=x; yPos=y;};

~Snake(){;};

int length, size, color, speed, xPos, yPos;

int borderColor, snakeColor, scr, stIndex, dlIndex;

int objX, objY;

int xArr[8000];

int yArr[8000];

int moveXRight();

int moveXLeft();

int moveYUp();

int moveYDown();

void store(void);

void erase(void);

void init(void);

void test(int x);

void putAnObject(void);

void getAkey(void);

int score(void);

void gameOver();

void restart();

void level(int x);

};

int Snake::moveXRight()

{

for(int i=0 ; i<800; i++)

{

setcolor(4);

store();

circle(xPos++,yPos,4);

delay(speed);

erase();

test(1);

if(xPos % 10 ==0)

if (kbhit())

break;

}

return xPos;

}

int Snake::moveXLeft(void)

{

for(int i=0 ; i<800; i++)

{

setcolor(4);

store();

circle(xPos--,yPos,4);

delay(speed);

erase();

test(2);

if(xPos % 10 ==0)

if (kbhit())

break;

}

return xPos;

}

int Snake::moveYUp()

{

for(int i=0 ; i<800; i++)

{

setcolor(4);

store();

circle(xPos,yPos--,4);

delay(speed);

erase();

test(3);

if(yPos % 10 ==0)

if (kbhit())

break;

}

return yPos;

}

int Snake::moveYDown()

{

for(int i=0 ; i<800; i++)

{

setcolor(4);

store();

circle(xPos,yPos++,4);

delay(speed);

erase();

test(4);

if(yPos % 10 ==0)

if (kbhit())

break;

}

return yPos;

}

void Snake::store()

{

xArr[stIndex]= xPos;

yArr[stIndex]= yPos;

stIndex++;

if (stIndex == size)

stIndex=0;

}

void Snake::erase()

{

setcolor(0);

if(dlIndex-length >=0)

circle(xArr[dlIndex-length],yArr[dlIndex-length],4);

else

circle(xArr[size-abs(dlIndex-length)], yArr[size-abs(dlIndex-length)],4);

dlIndex++;

if(dlIndex == size)

dlIndex =0;

}

void Snake::getAkey(void)

{

char key;

for(int k=0 ;k<3 ;k++)

{

k=1; // infinite loop

key = getch();

if (key == 'M')

xPos = moveXRight();

if(key == 'K')

xPos = moveXLeft();

if (key == 'H')

yPos = moveYUp();

if(key == 'P')

yPos = moveYDown();

if(key == 'q' || key == 'Q')

exit(1);

}

}

void Snake::test(int x)

{

int tst,a;

switch (x) {

case 1: { //when moving right

tst = getpixel(xPos+4,yPos);

if(xPos == objX\*10 &&yPos == objY\*10)

a=score();

level(a);

if(tst== borderColor || tst == snakeColor)

gameOver(); break;} // Game over

case 2: { //when moving left

tst = getpixel(xPos-4,yPos);

if(xPos == objX\*10&&yPos == objY\*10)

a=score();

level(a);

if(tst== borderColor || tst == snakeColor)

gameOver(); break;} // Game over

case 3: { //when moving up

tst = getpixel(xPos,yPos-4);

if(xPos == objX\*10 &&yPos == objY\*10)

a=score();

level(a);

if(tst== borderColor || tst == snakeColor)

gameOver(); break;} // Game over

case 4: { //when moving down

tst = getpixel(xPos,yPos+4);

if(xPos == objX\*10 &&yPos == objY\*10)

a=score();

level(a);

if(tst== borderColor || tst == snakeColor)

gameOver();break;} // Game over

} // end of switch

}

void Snake::level(int x)

{

settextstyle(0,0,2);

if(x==100)

{ ellipse(230,45,30,30,100,20);

setfillstyle(1,YELLOW);

fillellipse(300,45,100,20);

setcolor(BLUE);

outtextxy(250,40,"LEVEL 2");

sound(250);

delay(1000);

nosound();

speed=speed-3;

}

if(x==200)

{ ellipse(230,45,30,30,100,20);

setfillstyle(1,YELLOW);

fillellipse(300,45,100,20);

setcolor(RED);

outtextxy(250,40,"LEVEL 3");

sound(250);

delay(1000);

nosound();

speed=speed-3;

}

if(x==300)

{ ellipse(320,240,60,60,120,50);

setfillstyle(9,BLUE);

fillellipse(320,240,120,50);

delay(300);

setcolor(WHITE);

settextstyle(0,0,3);

outtextxy(240,230,"YOU WON");

delay(5000);

exit(1);

}

}

void Snake::init(void)

{

borderColor = YELLOW;

setcolor(RED);

settextstyle(0, 1, 6);

outtextxy(50,10,"RED SNAKE");

gotoxy(14,6);

cout<<"SCORE : 0";

setcolor(GREEN);

settextstyle(0,0,2);

ellipse(230,45,30,30,100,20);

setfillstyle(1,YELLOW);

fillellipse(300,45,100,20);

outtextxy(250,40,"LEVEL 1");

settextstyle(7,0,1);

setcolor(GREEN);

outtextxy(255,450,"\*\*\*Q=Quit \*\*\*");

setcolor(borderColor);

rectangle(100,100,540,380);

rectangle(105,105,535,375);

rectangle(110,110,530,370);

scr=0;

dlIndex=0;

stIndex=0;

speed=12;

snakeColor = RED;

size = 7999;

length = 50;

for(int i=0 ;i<8000; i++)

{

xArr[i] = 0;

yArr[i] = 0;

}

}

void Snake::putAnObject()

{

int test2=0;

for(int i=0 ;i<1000 ; i++)

{

objX = random(41)+12 ;

objY = random(25)+12 ; // the less minus 10

test2 = getpixel(objX\*10,objY\*10);

setcolor(WHITE);

if(test2 != snakeColor)

{ circle(objX\*10,objY\*10,4);

setfillstyle(2 ,WHITE);

fillellipse(objX\*10,objY\*10,4,4);

break;

}

}

}

void Snake::gameOver()

{

setcolor(WHITE);

ellipse(320, 240,30,30, 120, 25);

setfillstyle(9 ,BLUE);

fillellipse(320, 240, 120, 25);

settextstyle(0,0,2);

setcolor(RED);

outtextxy(250,232,"GAME OVER ");

getch();

getch();

restart();

}

void Snake::restart()

{

cleardevice();

xPos=320;yPos=240;

init();

randomize();

putAnObject();

getAkey();

}

int Snake::score()

{

scr+=10;

gotoxy(14,6);

cout<<"SCORE : ";

cout<<scr;

sound(2500);

delay(80);

nosound();

length+=20;

putAnObject();

return (scr);

}

void main(void) /\* MAIN \*/

{

int gdriver = DETECT, gmode, errorcode;

initgraph(&gdriver, &gmode, "c:\\tc\\bgi");

randomize();

Snake snake(320,240);

snake.init();

snake.putAnObject();

snake.getAkey();

getch();

closegraph();

}

**FUNCTIONS USED IN THE PROGRAM**

**gotoxy**

Positions cursor in text window

**Defined in:** *<CONIO.H>*

**Declaration:** *void gotoxy(int x, int y);*

**Remarks:**

* gotoxy moves the cursor to the given position in the current text window.
* If the coordinates are invalid, the call to gotoxy is ignored.

Example of invalid coordinates:

gotoxy(40,30) /\* (35,25) = window's bottom right position \*/

**Return Value:** None

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**delay**

Suspends execution for interval (milliseconds)

**Defined in:** *<DOS.H>*

**Declaration**: *void delay(unsigned milliseconds);*

**Remarks:**

* With a call to delay, the current program is suspended from execution for the time specified by the argument milliseconds.
* delay is accurate to one millisecond.

**Return Value**: None

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**randomize**

Macro that initializes random number generator

**Defined in:** *<STDLIB.H>*

**Declaration:** *void randomize(void);*

**Remarks:**

* randomize initializes the random number generator with a random value.
* Because randomize is implemented as a macro that calls the time function prototyped in *TIME.H*, you should include *TIME.H* when you use this routine.

**Return Value:** None

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**cleardevice**

Clears the graphics screen

**Defined in:** *<GRAPHICS.H>*

**Declaration:**  *void far cleardevice (void);*

**Remarks:**

* cleardevice erases the entire graphics screen and moves the CP (current position) to home(0,0).
* (Erasing consists of filling with the current background color.)

**Return Value:**  None

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**outtext, outtextxy**

* *outtext* displays a string in the viewport (graphics mode)
* *outtextxy* displays a string at the specified location (graphics mode)

**Defined in*:*** *<GRAPHICS.H>*

**Declaration:**

* *void far outtext(char far \*textstring);*
* *void far outtextxy(int x, int y, char far \*textstring);*

**Remarks:**

*outtext* and *outtextxy* display a text string, using the current justification settings and the current font, direction, and size.

* *outtext* outputs text string at the current position (CP)
* *outtextxy* displays text string in the viewport at the position (x, y)

If a string is printed with the default font using *outtext* or *outtextxy*, any part of the string that extends outside the current viewport is truncated.

*outtex*t and *outtextxy* are for use in graphics mode; they will not work in text mode.

**Return Value**: None

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**setcolor**

*setcolor* sets the current drawing color

**Defined in:** *<GRAPHICS.H>*

**Declaration**: *void far setcolor(int color);*

**Remarks:**

* *setcolor* sets the current drawing color to color, which can range from 0 to *getmaxcolor*.
* To select a drawing color with *setcolor*, you can pass either the color number or the equivalent color name.
* The drawing color is the value that pixels are set to when the program draws lines, etc.

**Return Value:** None

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**settextstyle**

Sets the current text characteristics

**Defined in:** *<GRAPHICS.H>*

**Declaration:** *void far settextstyle(int font, int direction, int charsize);*

**Remarks:**

* *settextstyle* sets the text font, the direction in which text is displayed, and the size of the characters.
* A call to *settextstyle* affects all text output by *outtext* and *outtextxy*.

**Return Value**: None

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ellipse, fillellipse**

* *ellipse* draws an elliptical arc
* *fillellipse* draws and fills an ellipse

**Defined in:** *<GRAPHICS.H>*

**Declaration:**

* *void far ellipse(int x, int y, int stangle, int endangle, int xradius, int yradius);*
* *void far fillellipse(int x, int y, int xradius, int yradius);*

**Remarks:**

* *ellipse* draws an elliptical arc in the current drawing color.
* *fillellipse* draws an ellipse, then fills the ellipse with the current fill color and fill pattern.

**Argument**

*(x,y)* - Center of ellipse

*xradius* - Horizontal axis

*yradius* - Vertical axis

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**setfillstyle**

Sets the fill pattern and color

**Defined in:** *<GRAPHICS.H>*

**Declaration:** *void far setfillstyle(int pattern, int color);*

**Remarks:** *setfillstyle* sets the current fill pattern and fill color.

**Return Value**: None

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**circle**

*circle* draws a circle

**Defined in:** *<GRAPHICS.H>*

**Declaration:** *void far circle(int x, int y, int radius);*

**Remarks:** *circle* draws a circle in the current drawing color.

**Return Value:** None

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**getpixel**

*getpixel* gets the color of a specified pixel

**Defined in:** *<GRAPHICS.H>*

**Declaration:** *unsigned far getpixel(int x, int y);*

**Remarks**: *getpixel* gets the color of the pixel located at *(x,y)*.

**Return Value:** *getpixel* returns the color of the given pixel.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**kbhit**

Checks for currently available keystrokes.

**Defined in:** *<CONIO.H>*

**Declaration**: *int kbhit(void);*

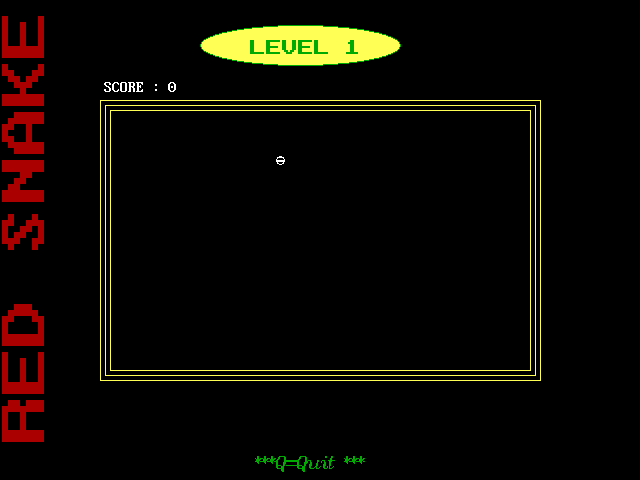
**Remarks:**

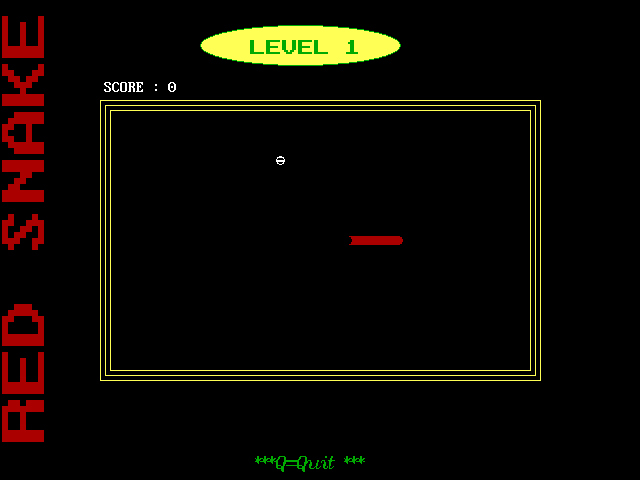
* *kbhit* checks to see if a keystroke is currently available.
* Any available keystrokes can be retrieved with *getch* or *getche*.

**Return Value:**

* On success (if a keystroke is available), returns a non-zero integer
* If a keystroke is not available, returns 0.

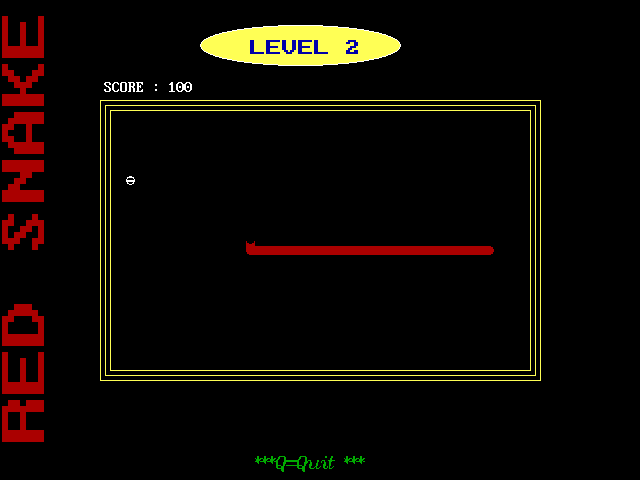
**OUTPUT**

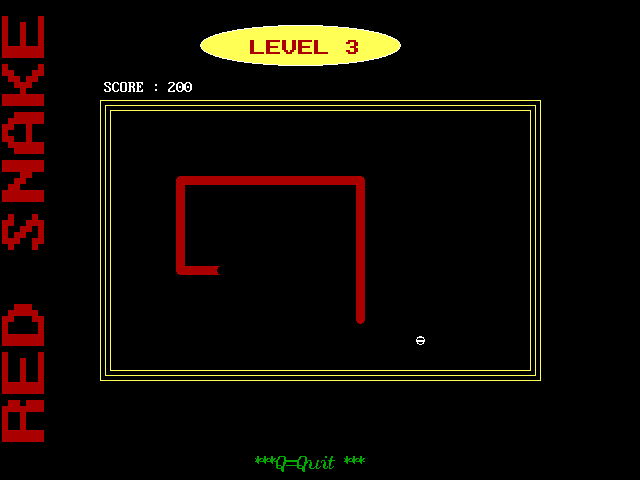
****



Level 1

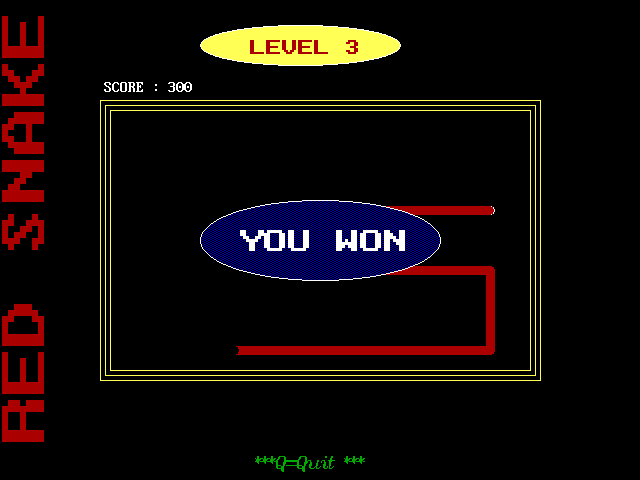
Initializing The Game



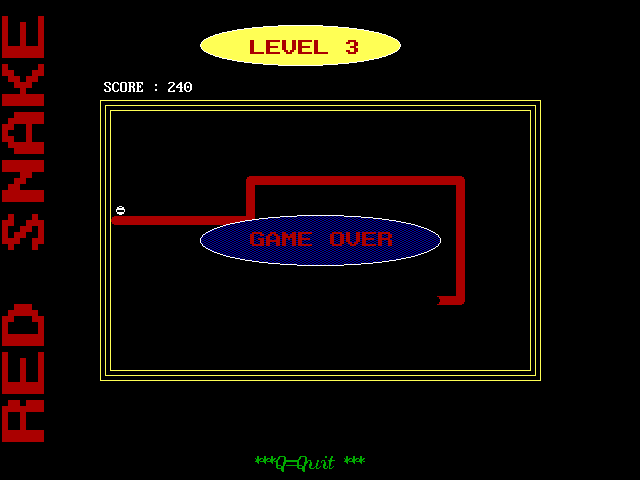


Level 3 After Reaching 200 Points

Level 2 After Reaching 100 Points



Completing The Game After Reaching 300 Points



Game Over If The Snake Touches Itself Or The Boundary

**BIBLIOGRAPHY**

* www.cprogramming.com
* www.mycplus.com
* www.devmaster.net
* Object oriented programming by E. Balaguruswamy
* http://www.simonhuggins.com/courses/cbasics/course\_notes/snake.htm
* http://answers.yahoo.com/question/index?qid=20070424153140AAeZdSO